Görev-16

Amaç:

Bu görevde şimdiye kadar yapılan projelerdeki business işlemlerini servisler aracılığıyla yapılacak şekilde düzenlenerek domain servislerinin kullanıldığı, katmanlı mimariye uygun, modüler bir proje yapısı oluşturulması amaçlanmıştır. Servisler ve web uygulamaları arasındaki veri aktarımı DTO modelleri aracılığı ile olacağı için ilgili servis metotlarının dönüş tipi ve parametrelerinde kullanılacak DTO modelleri de oluşturulacaktır.

Yapılacaklar:

- MVC projelerindeki action'larda, API'lere istek yapan kodlar için kullanılan HttpClient class'ını dependency injection ile singleton olarak kullanacak şekilde güncelleyin. Bunun için Microsoft tarafından önerilen yöntem, HttpClient'ın *IHttpClientFactory* interface'i aracılığı ile kullanılmasıdır (Bkz: Referans 1)
- Solution altına "**App.Services**" isimli bir class library oluşturun. Bu class library projesi servis interface ve class'larını içerecektir.
- Oluşturduğunuz bu class library içerisine "**Abstract**" ve "**Concrete**" isimli iki klasör oluşturun. Abstract klasörü servis interface'lerini, Concrete klasörü de bu interface'lerin implementasyonları olan class'ları içerecektir.
- MVC ve servis projelerine **Ardalis.Result** NuGet paketini ekleyin. Oluşturulacak servis metotlarının döndürdüğü değerler bu kütüphane içerisindeki *Result* tipinde olacaktır. (Bkz: Örnek, Referanslar 2, 3)
- Solution altına **"App.Models.DTO"** isimli bir class library projesi ekleyin. Bu proje, DTO modellerini içerecektir. (Bkz: Referanslar 4, 5)
- **App.Services** projesinin proje referanslarına **App.Models.DTO** projesini ekleyerek ileride burada oluşturulacak DTO modellerini servislerin kullanabilmesini sağlayın.
- MVC projelerindeki action metotlarındaki business kodları için **App.Services** projesindeki Abstract klasörü içerisinde servis interface'lerini, Concrete klasörü içerisine de bu interface'leri implemente eden somut servis sınıflarını oluşturun. Servis interface'lerini oluştururken benzer domain işlemlerini yapan kodları gruplayarak aynı interface içerisinde toplamaya özen gösterin. Bunun için aşağıda verilen **Yetkiler ve Modüller** tablosundaki modül sütununa göre interface'ler oluşturabilirsiniz. Servis metotlarının aldığı ve geri döndürdüğü DTO modellerini de App.Models.DTO class library projesi içerisine oluşturun. (Bkz: Örnek)
- Abstract ve concrete servisler ve DTO'lar oluşturulup proje referansları düzenlendikten sonra MVC projelerine bu servisleri DI (dependency injection) yapılmak üzere ekleyin. Bunun için ilgili projelerdeki Program.cs dosyalarında değişiklik yapılmalıdır.

- MVC controller action'larındaki işlemlerin, oluşturduğunuz servis interface'leri üzerinden yapılması için gerekli değişiklikleri yapın. Servis metotlarından dönen *Result* tipindeki sonuçları, ilgili action'ın dönüş tipine uygun olarak kullanın.
- API metodlarındaki kodlar için de benzer adımları uygulayarak abstract ve concrete servisleri ile DTO modellerini oluşturun, dependency injection için gerekli güncellemeleri yapın ve servislerden gelen Result tipindeki sonuçları, ilgili action'ın dönüş tipine uygun şekilde kullanın.

Örnek (ASP.NET Core 8.0):

App.Models.DTO projesi içerisinde:

```
// NewOrderRequest.cs dosyası:
public class NewOrderRequestDto {
        public List<NewOrderItemDto> Items { get; set; }
        public string DeliveryAddress { get; set; }
}

// NewOrderItemDto.cs dosyası:
public class NewOrderItemDto {
        public int ProductId { get; set; }
        public byte Quantity { get; set; }
}

// NewOrderResponseDto.cs dosyası:
public class NewOrderResponseDto {
        public int OrderId { get; set; }
}
```

App.Services projesindeki Abstract klasörü içerisinde:

```
// IOrderService.cs dosyası:
using Ardalis.Result;
public interface IOrderService {
        Task<Result<NewOrderResponseDto>> PlaceOrderAsync(string jwt,
                NewOrderRequestDto newOrderRequest);
        Task<Result<MyOrdersResponseDto>> GetMyOrders(string jwt);
        Task<Result<MyOrderDetailResponseDto>> GetMyOrderDetails(string jwt,
                MyOrderDetailRequestDto myOrderDetailRequest);
}
```

App.Services projesindeki Concrete klasörü içerisinde:

```
// OrderService.cs dosyası:
using System.Net.Http;
using Ardalis.Result;

public class OrderService : IOrderService {
        private readonly HttpClient client;
        public OrderService(IHttpClientFactory httpClientFactory) {
                client = httpClientFactory.CreateClient("DataApi");
        }

        // JWT header'ını ekleyerek istek yapan yardımcı metot (ayrı bir servis şeklinde yapılarak DI ile de kullanılabilir):
        protected async Task<HttpResponseMessage> SendApiRequestAsync(string apiRoute,
                HttpMethod method,
                string jwt,
                object payload = null)
        {
                var httpRequestMessage = new HttpRequestMessage(
                        method,
                        apiRoute)
                        {
                            Headers =
                            {
                                { HeaderNames.Authorization, $"Bearer {jwt}" }
                            }
                        };
                if (payload is not null) {
                        httpRequestMessage.Content = JsonContent.Create(payload);
                }

                return await client.SendAsync(httpRequestMessage);
        }

        public async Task<Result<NewOrderResponseDto>> PlaceOrderAsync(string jwt,
                NewOrderRequestDto newOrderRequest)
        {
                var response = await SendApiRequestAsync("api/order",
                        HttpMethod.Post,
                        jwt,
                        newOrderRequest);
```

```
                if (!response.IsSuccessStatusCode) {
                        // Başarısız işlemler için uygun erken dönüş (early return)
                        // ör 1: Result.Unauthorized();
                        // ör 2: Result.NotFound();
                }
                // Başarılı işlem:
                var newOrderResponse = await
                response.Content.ReadFromJsonAsync<NewOrderResponseDto>();

                return Result.Success(newOrderResponse);
        }

        // interface implementasyonu devamı ...
}
```

Yetkiler ve Modüller Tablosu (Görev-14 ve 15'deki ile aynı):

No	Proje	Modül	User Story	Role: Buyer	Role: Seller	Role: Admin
1	e-ticaret	Auth	Kullanıcı sisteme kayıt olabilecektir	✔	✘	✘
2	e-ticaret, admin	Auth	Kullanıcı sisteme email ve şifre ile giriş yapabilecektir	✔	✔	✔
3	e-ticaret, admin	Auth	Kullanıcı sistemden çıkış yapabilecektir	✔	✔	✔
4	e-ticaret, admin	Profile	Kullanıcı kendi profil sayfasını görüntüleyebilecektir	✔	✔	✔
5	e-ticaret	Profile	Kullanıcı, satıcı olmak isteğini talep edebilecektir	✔	✘	✘
6	e-ticaret	Cart	Kullanıcı stokta varsa ürünü sepetine ekleyebilecektir	✔	✔	✘
7	e-ticaret	Cart	Kullanıcı ürünü sepettinden çıkarabilecektir	✔	✔	✘
8	e-ticaret	Cart	Kullanıcı sepetteki ürününün adedini güncelleyebilecektir	✔	✔	✘
9	e-ticaret	Cart	Kullanıcı sepetinin detaylarını görüntüleyebilecektir	✔	✔	✘
10	e-ticaret	Order	Kullanıcı ödeme yaptıktan sonra adres bilgisi girecek, ardından siparişi otomatik oluşacaktır	✔	✔	✘
11	e-ticaret	Order	Kullanıcı siparişlerinin listesini görebilecektir	✔	✔	✘
12	e-ticaret	Order	Kullanıcı tek bir siparişinin detaylarını görebilecektir	✔	✔	✘
13	admin	Category	Kullanıcı sisteme yeni bir kategori ekleyebilecektir	✘	✘	✔
14	e-ticaret	Category	Kullanıcı bir kategori altındaki ürünleri listeyebilecektir	✔	✔	✘
15	e-ticaret	Product	Kullanıcı sisteme yeni ürün ekleyebilecektir	✘	✔	✘
16	e-ticaret	Product	Kullanıcı satıştaki kendi ürünlerini listeleyebilecektir	✘	✔	✘
17	e-ticaret	Product	Kullanıcı sistemdeki mevcut kendi ürününün fiyatını güncelleyebilecektir	✘	✔	✘
18	e-ticaret	Product	Kullanıcı sistemdeki mevcut kendi ürününün stoğunu güncelleyebilecektir	✘	✔	✘
19	e-ticaret	Product	Kullanıcı sipariş oluşturduktan sonra ürüne yorum ve yıldız verebilecektir	✔	✔	✘
20	e-ticaret	Product	Kullanıcı tek bir ürünün detaylarını görebilecektir	✔	✔	✘
21	e-ticaret	Product	Kullanıcı genel ürün araması yapabilecektir	✔	✔	✘
22	e-ticaret, admin	Product	Kullanıcı, ürünü pasif hale getirebilecektir	✘	✔	✔
23	e-ticaret	ProductComment	Kullanıcı satın aldığı ürünlere yorum ve yıldız verebilecektir	✔	✔	✘
24	admin	ProductComment	Kullanıcı, ürün yorumlarını herkesin görebilmesi için onaylayabilecektir	✘	✘	✔
25	admin	User	Kullanıcı, sistemdeki tüm kullanıcıları listeleyebilecektir	✘	✘	✔
26	admin	User	Kullanıcı, diğer kullanıcıları aktif/pasif yapabilecektir	✘	✘	✔
27	admin	User	Kullanıcı, satıcı olma taleplerini onaylayabilecektir	✘	✘	✔

Referanslar:

1. Microsoft: Make HTTP requests using IHttpClientFactory in ASP.NET Core
2. Github: Ardalis.Result
3. Ardalis.Result Kullanımı: Getting Started With Ardalis.Result
4. Wikipedia: Data transfer object
5. Microsoft: Create Data Transfer Objects (DTOs)

Teslim: Proje klasörünü zip'leyip classroom'daki Görev-16 paylaşımına ek olarak ekleyin.